Tarea Complementaria 4

📘 Investigación de conceptos sobre XML y sus analizadores

---

1. Introducción general

En el módulo de Acceso a Datos, uno de los bloques más importantes es el trabajo con documentos XML. Aunque hoy en día se usan formatos más ligeros como JSON, XML sigue presente en muchísimas aplicaciones, sobre todo en entornos empresariales, sistemas antiguos, comunicación entre servicios o configuraciones complejas.
Para poder manejar XML desde Java necesitamos entender bien **cómo se interpreta un XML**, qué tipos de analizadores existen y qué implica usar uno u otro.

A continuación explico los conceptos más importantes y, sobre todo, qué papel tiene cada uno en la práctica.

---

2. Conceptos

XML

XML es un formato de texto que sirve para guardar información de forma estructurada. No es un lenguaje diseñado para "mostrar" cosas, sino para **describir datos**. La gracia de XML es que tú defines tus propias etiquetas, y eso lo convierte en un formato muy flexible. Al final, lo que conseguimos es una especie de árbol jerárquico formados por nodos.

Un detalle importante: XML es independiente de plataforma, fácil de leer por personas y también por máquinas.

Metalenguaje

Se dice que XML es un metalenguaje porque no es un lenguaje "cerrado"; es un lenguaje que permite crear **otros lenguajes** encima. Por ejemplo, formatos como SVG, XHTML o los layouts de Android se construyen usando XML. Tú eliges las etiquetas y decides la estructura.

Parser / Analizador sintáctico

Un parser es simplemente la pieza del software que se encarga de **leer un XML y entender su estructura**.
Es decir:

- revisa las etiquetas,
- las abre, las cierra,
- lee los atributos,
- detecta errores,
- y transforma todo eso en algo que el programa pueda manejar.

En Java tenemos dos formas principales de trabajar con XML: DOM y SAX.

Handler (en SAX)

Cuando usamos SAX, no obtenemos un árbol en memoria. SAX funciona por “eventos”: cuando encuentra la apertura de una etiqueta llama a un método, cuando la cierra llama a otro, cuando encuentra texto llama a otro…

Ese conjunto de métodos lo defines tú mismo en una clase llamada **handler**. Ahí decides qué hacer en cada caso.

DOM

DOM funciona de forma muy distinta. En vez de leer el XML por partes, **carga el XML entero en memoria** y lo representa como un árbol de nodos.

¿Ventaja? Que puedes recorrerlo como quieras, arriba, abajo, derecha, izquierda… modificarlo, añadir nuevos nodos, eliminar, guardar, etc.

¿Problema? Que si el archivo XML es grande, el consumo de memoria se dispara. DOM está pensado para XML pequeños o medianos.

SAX

SAX es todo lo contrario que DOM. No carga nada en memoria, sino que va leyendo línea a línea, disparando eventos.

Es muy rápido, consume muy poca memoria y es ideal para XML enormes. Pero tiene una limitación importante: **no puedes modificar el XML**, ni navegar hacia atrás, ni saltar a una parte concreta. Solo se lee en orden.

JAXB (Jakarta XML Binding)

JAXB es una solución completamente distinta. En vez de trabajar directamente con nodos o eventos, JAXB permite convertir XML directamente en objetos Java (“unmarshalling”) y convertir objetos Java en XML (“marshalling”).
Lo habitual es usar anotaciones como `@XmlRootElement` para marcar qué representa cada clase.

Es una manera muy cómoda de trabajar si el XML representa entidades del mundo real (personas, productos, libros, etc.). Es decir, no piensas en etiquetas, sino en objetos Java normales.

Desde Java 11 ya no viene incluido en el JDK, así que hay que añadir las dependencias manualmente.

Binding (vinculación)

Cuando hablamos de binding nos referimos precisamente a ese proceso de “vincular” datos XML con objetos Java. JAXB automatiza esto al máximo. Tú creas las clases Java y JAXB se encarga de unirlas con el contenido del XML.

Navegación bidireccional

Este concepto solo tiene sentido con DOM. Como DOM genera un árbol completo en memoria, puedes moverte por él de cualquier forma: subir al padre, recorrer los hijos, ver los hermanos, etc.

Con SAX esto es imposible; solo lees en una dirección.

---

3. Comparación entre DOM, SAX y JAXB

Para que quede más claro, resumo sus usos reales en situaciones del día a día:

DOM

- Lo usaría cuando necesito modificar el XML, guardarlo de nuevo, o navegarlo en profundidad.
- Funciona genial con documentos pequeños o medianos.
- No lo usaría jamás con XML enormes.

SAX

- La mejor opción cuando necesitas leer XML enormes o extraer datos de forma muy rápida.
- No sirve para editar.
- Tampoco permite acceder a nodos anteriores.

JAXB

- Útil cuando estás trabajando en aplicaciones que manejan datos estructurados, donde el XML representa entidades de un modelo.
- Muy limpio y mantenible.
- No es buena idea para documentos gigantes.

---

4. Preguntas

1. DOM carga todo el XML en memoria. ¿Verdadero o falso?

Es totalmente verdadero. Esa es precisamente su filosofía: coger el XML entero y convertirlo en un árbol que puedas recorrer.
Esto hace que DOM sea muy cómodo, pero al mismo tiempo muy pesado si el archivo es grande.

---

2. ¿Qué usarías para procesar un XML de 5 GB?

Aquí no hay duda: **SAX**.
Cualquier cosa que cargue el documento completo moriría al instante, porque 5 GB de XML convertirían tu RAM en puré. SAX lo procesa como si fuera un streaming, lo que permite manejar volúmenes gigantescos sin problema.

---

3. ¿Qué hay que añadir en Java 17 para usar JAXB?

Hace unos años venía integrado en el JDK, pero a partir de Java 11 lo sacaron.
Para usar JAXB en Java moderno hace falta añadir las dependencias (las típicas de `jakarta.xml.bind` y `jaxb-runtime`).
Sin eso, el código que antes funcionaba dará error porque las clases de JAXB simplemente no existen en el JDK.

---

4. Diferencia entre “bien formado” y “válido”

Bien formado significa que el documento cumple las reglas básicas del XML: etiquetas cerradas, estructura correcta, un nodo raíz, atributos con comillas…

Válido significa que, además, sigue lo que dice una DTD o un XSD.
Es como decir:

- Bien formado → “está escrito correctamente”.
- Válido → “además cumple las normas que se le exigen”.

---

5. ¿Qué utilizarías en varios escenarios?

a) Procesar un fichero enorme de logs en XML
SAX sin duda. Es rápido, consume poca memoria y no necesitas editar.

b) Editar un XML pequeño para modificar datos y guardarlo
Aquí DOM es perfecto porque te permite navegar y manipular la estructura fácilmente. JAXB también serviría, pero sería más complejo si solo quieres hacer cambios puntuales y no mapear todo el documento a objetos.

---

5. Conclusión general

XML puede parecernos un formato “viejo”, pero sigue siendo muy utilizado. La clave está en saber elegir la herramienta adecuada:

- Si quieres leer documentos enormes: SAX.
- Si quieres manipularlos como si fueran un árbol: DOM.
- Si quieres trabajar con objetos Java directamente: JAXB.

Dominar estas tres herramientas te permite manejar cualquier XML que aparezca durante el ciclo, exámenes o en tu futuro trabajo.
